消息中心接入文档

本文档描述通过多种方式接入消息中心发送 SMS 短信

1、Java 端接入

使用 openfeign 客户端方式接入消息中心，因此需要引入 spi 客户端 jar 包，以及 openfeign 相关 jar，如果之前工程已经引入 openfeign 的相关 jar 包，则无需再次引入。

java 客户端 SDK（message-spi）封装的能力：

（1）调用消息中心 http 接口；

（2）封装消息中心 http 接口签名；

（3）手机号码加密传输；

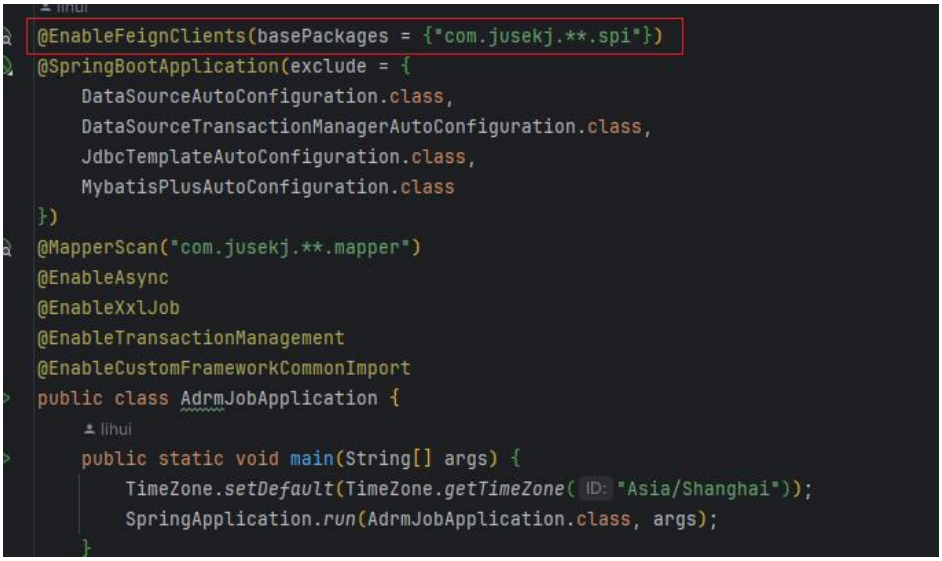
（4）封装回调回执接口；

1.1、引入 maven 依赖

```
<dependency>
    <groupId>com.jusekj.message</groupId>
    <artifactId>message-spi</artifactId>
    <version>0.0.2-SNAPSHOT</version>
</dependency>

<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-openfeign</artifactId>
    <version>2.2.5.RELEASE</version>
</dependency>

<dependency>
    <groupId>com.netflix.feign</groupId>
```

```
    <artifactId>feign-slf4j</artifactId>
    <version>8.14.4</version>
</dependency>
```

1.2、启用 openfeign，在 spring boot 启动类添加@EnableFeignClients 注解



```java
@EnableFeignClients(basePackages = {"com.jusekj.**.spi"})
@SpringBootApplication(exclude = {
    DataSourceAutoConfiguration.class,
    DataSourceTransactionManagerAutoConfiguration.class,
    JdbcTemplateAutoConfiguration.class,
    MybatisPlusAutoConfiguration.class
})
@MapperScan("com.jusekj.**.mapper")
@EnableAsync
@EnableXxlJob
@EnableTransactionManagement
@EnableCustomFrameworkCommonImport
public class AdrmJobApplication {
    lihui
    public static void main(String[] args) {
        TimeZone.setDefault(TimeZone.getTimeZone( ID: "Asia/Shanghai"));
        SpringApplication.run(AdrmJobApplication.class, args);
    }
```

1.3、配置说明

```
juse:
  server:
    spi:
      # 消息中心服务环境地址，msgdev 开发环境，msgtest 测试环境，msg 正式环境
      msg: https://msgdev.gooeto.com/api
  message:
    # 默认值是 message，无需配置，消息中心自身服务的编码
    appKey: message
    # 应用编号，向消息中心系统管理员申请
    appId: adrm
    # 应用密钥，向消息中心系统管理员申请
    appSecret: r6duQGOuwDw4QeF/TJi6tw==
    # 默认值/api，无需配置，消息中心忽略的路径（nginx 代理转发的 path，消息中心后端接收不到此路径）
```

ignorePath: /api

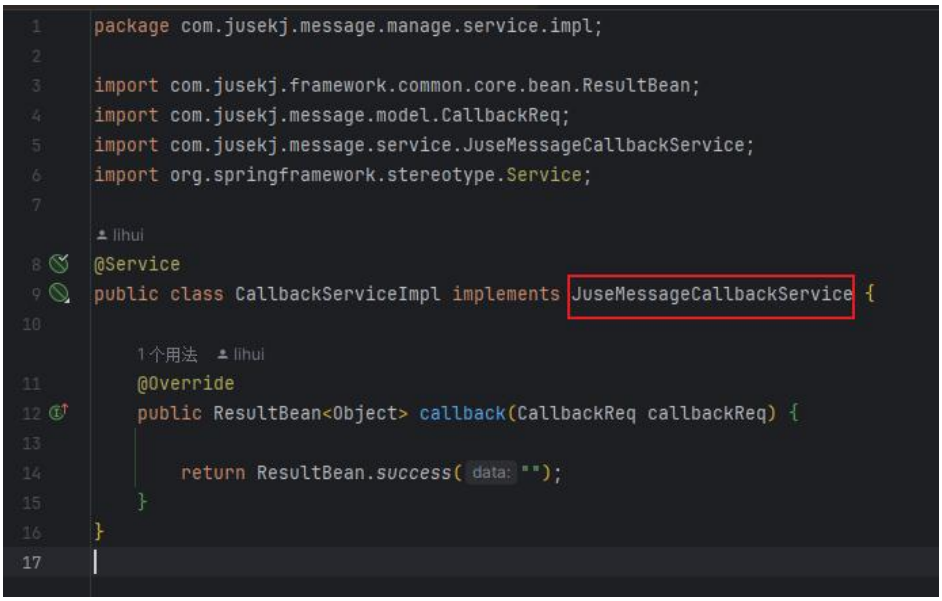
1.4、发送短信

```java
@Slf4j
@SpringBootTest
@RunWith(SpringRunner.class)
public class MessageTest {

    2个用法
    @Resource
    private MessageServiceSpi messageServiceSpi;


    @Test
    public void test(){
        // 发送模板短信
        sendMsg();

        // 发送任务计划短信
        sendTaskMsg();
    }

    1个用法
    private void sendTaskMsg(){
        SendSmsTaskReqDto sendSmsTaskReqDto = new SendSmsTaskReqDto();
        sendSmsTaskReqDto.setAppId("adrm");
        sendSmsTaskReqDto.setName("测试批量发送");
        sendSmsTaskReqDto.setTemplateId("MSG_2024090901");
        sendSmsTaskReqDto.setSignName("登录验证");
        sendSmsTaskReqDto.setPlanTime(null);
        sendSmsTaskReqDto.setPhones("15837390237");
        Map<String, Object> paramMap = new HashMap<>();
        paramMap.put("param1", "888999");
        sendSmsTaskReqDto.setParams(paramMap);

        ResultBean<SendSmsTaskRspDto> resultBean = messageServiceSpi.sendSmsTask(sendSmsTaskReqDto);
        log.info(JSONUtil.toJsonStr(resultBean));
    }
```

```java
    1个用法
    private void sendMsg(){
        SendSmsReqDto sendSmsReqDto = new SendSmsReqDto();
        sendSmsReqDto.setTemplateId("MSG_2024090901");
        sendSmsReqDto.setSignName("登录验证");
        sendSmsReqDto.setPhone("15837390237");
        sendSmsReqDto.setAppId("adrm");
        Map<String, Object> paramMap = new HashMap<>();
        paramMap.put("param1", "888999");
        sendSmsReqDto.setParams(paramMap);
        ResultBean<SendSmsRspDto> resultBean = messageServiceSpi.sendSms(sendSmsReqDto);
        log.info(JSONUtil.toJsonStr(resultBean));
    }
}
```

1.4、接收回调回执

SDK 封装了回调接口，接收服务只需实现 JuseMessageCallbackService 接口，即可收到短信发送状态回执



```java
package com.jusekj.message.manage.service.impl;

import com.jusekj.framework.common.core.bean.ResultBean;
import com.jusekj.message.model.CallbackReq;
import com.jusekj.message.service.JuseMessageCallbackService;
import org.springframework.stereotype.Service;

lihui
@Service
public class CallbackServiceImpl implements JuseMessageCallbackService {

    1个用法  lihui
    @Override
    public ResultBean<Object> callback(CallbackReq callbackReq) {

        return ResultBean.success( data: "");
    }
}
```

注意：接收服务需忽略/api/juse_message/callback 的认证

SDK 封装回调回执接口，便于后续增加签名验签逻辑，接收服务无需对接签名验签

2、HTTP 方式接入

使用 http 接口方式接入消息中心，需要根据消息中心规范实现接口签名，实现手机号码加密传输。

2.1、通用请求头参数

验签相关参数全部通过**请求头**传输，参数如下：

注意：表格中的”*“要替换成消息中心服务编码，默认值是字符串”message“，可编码设计为配置项，默认值是 message，可参考”1.3 配置说明“的 juse.message.appKey 配置项，默认值是：message，则请求头传参：X-message-Algorithm）：

请求头 name	是否必须	说明

请求头 name	是否必须	说明
X-*-Algorithm	是	X-*-Sign 指定的算法，可选范围目前只有 md5，后续如有必要会增添其他算法
X-*-Time	是	时间戳，请求时间，防止重放攻击，目前后台最多延时 15 分钟
X-*-AppType	是	终端应用类型，比如小程序设置为 1、WEB 端设置为 2、APP 端设置为 3，建议设置 2
X-*-AppID	是	应用唯一标识，用于判断是那个应用调用消息中心
X-*-Nonce	是	随机码，防止重放攻击，延时时间之内不能重复，客户端可以采用 UUID 生成
X-*-Sign	是	签名 Sign，具体说明看下文。

X-*-Sign 说明

参考计算方式为：

MD5(APPSECRET+路径+参数 +X-*-AppType +X-*-Time +X-*-Nonce)

指定的算法以下分别说明：

- **MD5：** 即为 **X-*-Algorithm** 指定算法 md5
- **APPSECRET**：签名秘钥，向消息中心管理员申请，可编码设计成配置项，类似像 1.3 配置说明的 juse.message.appSecret 配置，例如：r6duQGOuwDw4QeF/TJi6tw==
- **路径**：即为 url 中的 path 部分，要去除前缀/api 路径，因为消息中心部署 nginx 有转发前缀，消息中心服务获取不到这个前缀，则需要替换掉这个部分 path。例如完整 path 是/api/spi/sendSms，替换后的值是/spi/sendSms
- **参数**：post 请求方法，读取 body 流转换为 utf-8 字符串
- **X-*-AppType**：建议传 web 端，值为 2
- **X-*-Time**：请求头传输的时间，例如：1726631416
- **X-*-Nonce**：请求头传输的随机码，例如：6bab44992f9e4164812824e4aaf51628

例如：

APPSECRET+路径+参数 +X-*-AppType +X-*-Time +X-*-Nonce

r6duQGOuwDw4QeF/TJi6tw==/spi/sendSmsappId=adrm,params={param1=888999},phone=X4jPcOrqNtopIgUuoBe5Dw==,signName=登录验证,templateId=MSG_20240909011726630835b2d863e99be94f94a806c7f9b51e49e4

MD5(APPSECRET+路径+参数 +X-*-AppType +X-*-Time +X-*-Nonce)

1722092a360cc6c2478c75f0fbd94192

2.2、发送短信接口

接口说明：同步调用第三方短信服务商，异步回调通知调用结果

请求接口：https://msgdev.gooeto.com/api/spi/sendSms

请求方式：POST

数据格式：JSON

请求头参数：参考”2.1 通用请求头参数“

请求参数：

字段	类型	是否必填	说明
appId	String	是	应用编号
templateId	String	是	模板编号
signName	String	是	短信签名
phone	String	是	手机号码，使用 appSercet 作为密钥，通过 AES 加密传输
params	TreeMap<String, Object>	是	模板短信参数，移动 MAS 严格根据 key-value 传递的顺序替换模板字段，例如：{"param2","112233","param1":"445566"},替换参数是 param2 在 param1 前

举例：

```
{
    "appId": "adrm",
    "templateId":"MSG_2024090901",
    "signName": "登录验证",
    "phone":"15837390237",
    "params":{
        "param1": "888999"
    }
}
```

响应：

字段	类型	示例	说明
code	int	200	请求结果，非 200 为失败
mes	String		错误原因
data	Object		业务数据
requestId	String		请求编号

示例：


```
{
    "code": 400,
    "mes": "参数错误:X-message-AppID 参数为空",
    "requestId": null,
    "detail": "参数错误:X-message-AppID 参数为空",
    "httpStatusHint": 503,
    "showDetail": false,
    "ok": false
}
```


2.3、任务（批量）发送短信接口

接口说明：异步 job 定时发送，异步回调通知调用方结果，

请求接口：https://msgdev.gooeto.com/api/spi/sendSmsTask

请求方式：POST

数据格式：JSON

请求头参数：参考”2.1 通用请求头参数“

请求参数：

字段	类型	是否必填	说明
appId	String	是	应用编号
name	String	是	任务名称
templateId	String	是	模板编号
signName	String	是	短信签名
planTime	String	是	计划发送时间，例如：2024-09-18 10:00:00
phones	String	是	手机号码，多个使用逗号拼接，最多支持 1000，使用 appSercet 作为密钥，通过 AES 加密传输
params	Map<String, Object>	是	模板短信参数，移动 MAS 严格根据 key-value 传递的顺序替换模板字段，例如：{"param2", "112233","param1":"445566"},替换参数是 param2 在 param1 前

举例：

```
{
    "appId": "adrm",
    "name": "推广短信",
    "templateId":"MSG_2024090901",
    "signName": "登录验证",
    "phones":"15837390237,18566292142",
    "params":{
        "param1": "888999"
    }
}
```

响应：

字段	类型	示例	说明
code	int	200	请求结果，非 200 为失败
mes	String		错误原因
data	Object		业务数据
requestId	String		请求编号

示例：


```
{
    "code": 400,
    "mes": "参数错误:X-message-AppID 参数为空",
    "requestId": null,
    "detail": "参数错误:X-message-AppID 参数为空",
    "httpStatusHint": 503,
    "showDetail": false,
    "ok": false
}
```


2.4、接收回调回执

第三方服务需开发一个接口用于接收回执消息，接口规范如下：

请求方式：POST

数据格式：JSON

验签相关参数全部通过**请求头**传输，参数如下：

注意：表格中的”*“要替换成消息中心服务编码，默认值是字符串”message“，可编码设计为配置项，默认值是 message，可参考”1.3 配置说明“的 juse.message.appKey 配置项，默认值是：message，则请求头传参：X-message-Algorithm）：

请求头 name	是否必须	说明

请求头 name	是否必须	说明
X-*-Time	是	时间戳，请求时间，防止重放攻击，目前后台最多延时 15 分钟
X-*-Nonce	是	随机码，防止重放攻击，延时时间之内不能重复，客户端可以采用 UUID 生成
X-*-Sign	是	签名 Sign，具体说明看下文。

X-*-Sign 说明

参考计算方式为：

MD5(APPSECRET+X-*-Time +X-*-Nonce + 参数 appId + 参数 type)

指定的算法以下分别说明：

- MD5： 指算法 md5
- APPSECRET：签名秘钥，向消息中心管理员申请，可编码设计成配置项，类似像 1.3 配置说明的 juse.message.appSecret 配置，例如：r6duQGOuwDw4QeF/TJi6tw==
- X-*-Time：请求头传输的时间，例如：1726631416
- X-*-Nonce：请求头传输的随机码，例如：6bab44992f9e4164812824e4aaf51628
- 参数 appId：应用编号，下面请求参数的 appId 值，例如：adrm
- 参数 type：回调类型，下面请求参数的 type 值，例如：sms_report

例如：

APPSECRET+X-*-Time +X-*-Nonce + 参数 appId + 参数 type

r6duQGOuwDw4QeF/TJi6tw==17266314166bab44992f9e4164812824e4aaf51628adrmsms_report

MD5(APPSECRET+X-*-Time +X-*-Nonce + 参数 appId + 参数 type)

f04f1da7f21971881ef0ceee3eb1d455

请求参数：

字段	类型	说明
appId	String	应用编号
type	String	回调类型，sms_report 短信发送状态报告；sms_task_report 短信任务发送状态报告；sms_deliver 上行短信
extensions	Map<String, String>	扩展参数
sendSmsRecordDtoList	List<SendSmsRecordDto>	发送记录集合，type 为 sms_report 时不为空
sendRecordId	Integer, SendSmsRecordDto	发送记录编号
sendStatus	Integer, SendSmsRecordDto	发送状态：0-待发送，1-已发送，2-发送中， -1-发送失败
reportDate	String, SendSmsRecordDto	报告时间，yyyy-MM-dd HH:mm:ss
remark	String, SendSmsRecordDto	备注
smsDeliverDtoList	List<SmsDeliverDto>	上行短信集合，type 为 sms_task_report 时不为空
content	String, SmsDeliverDto	上行短信内容
mobile	String, SmsDeliverDto	上行手机号码，每批次携带一个号码
sendTime	String, SmsDeliverDto	上行短信发送时间，格式：yyyy-MM-dd HH:mm:ss
addSerial	String, SmsDeliverDto	上行服务代码
sendSmsTaskDto	SendSmsTaskDto	短信任务信息，type 为 sms_deliver 时不为空
sendTaskId	Long, SendSmsTaskDto	发送任务编号
totalCount	Integer, SendSmsTaskDto	号码总数

sendCount	Integer, SendSmsTaskDto	成功数量
failCount	Integer, SendSmsTaskDto	失败数量

举例：

```
{
    "appId": "adrm",
    "type": "sms_report",
    "extensions":{},
    "sendSmsRecordDtoList": [],
    "sendSmsTaskDto": {
                "sendTaskId": 1,
                "totalCount": 5,
                "sendCount": 3,
                "failCount": 2
    }
}
```

响应：

字段	类型	示例	说明
code	int	200	请求结果，非 200 为失败
mes	String		错误原因
data	Object		业务数据
requestId	String		请求编号

示例：

```
{
    "code": 400,
    "mes": "参数错误:X-message-AppID 参数为空",
    "requestId": null,
    "data": "参数错误:X-message-AppID 参数为空"
}
```